CIS 675 Fall 2015 Homework 5

Instructions: Submit your solutions in a pdf file to Blackboard by midnight Eastern time on **December 9, 2015**. Late submissions will not be accepted!

All answers must be proved: it is not sufficient to simply state the answer.

You may work with other students, but please note who you worked with on your submission. If you consult outside sources (such as Wikipedia), cite your source. All answers must be written in your own words.

Problem 1: The Hitting Set problem is as follows: Suppose we have a set of elements $V = \{v_1, ..., v_n\}$. Suppose we have a collection of sets $S_1, ..., S_k$, where each S_i is a set containing some of the v_j elements. A hitting set is a set H that is a subset of V, such that H contains at least one element from every S_i. Given some value b, we wish to find a hitting set with b or fewer elements, if it exists.

For example, suppose $S_1 = \{v_1, v_2, v_4\}, S_2 = \{v_2, v_3, v_5\}, S_3 = \{v_1, v_3, v_5\}$, and $b = 3$. Is there a hitting set with 3 or fewer elements? One example of a hitting set is $\{v_2, v_5\}$. This hitting set has size 2, so we have found a solution to the problem.

The Vertex Cover problem is as follows: Given a graph G and some value c, is there a set of c vertices in the graph such that every edge in the graph is adjacent to at least one of those c vertices (in other words, every edge must have at least one of its two endpoints included in the set of c vertices)? (Note that this problem definition is slightly different from what we did in class.)

Show that Hitting Set is NP-Complete by reducing the Vertex Cover problem to Hitting Set.

Solution: Suppose that we had an algorithm to solve Hitting Set. Then for every edge (u, v), create a set $\{u, v\}$. There is thus one set for every edge in the graph. Suppose we want to find a vertex cover with c or fewer vertices. Then find a hitting set over these sets that has c or fewer elements. This way, at least one vertex from each edge will be included.

Problem 2: Suppose we consider a variation of the SAT formula, in which we are given a Boolean formula and an integer g. In the original SAT problem, we needed to assign truth values to literals such that every clause in the formula was satisfied. In this problem, we only need to satisfy g clauses. Call this problem g-SAT. Show that g-SAT problem is NP-Complete by reducing SAT to g-SAT.

Solution: Suppose we have an algorithm to solve g-SAT. Then to solve a given SAT problem, simply set g to be the number of clauses in the problem.

Problem 3: In class, we covered the CLIQUE problem, in which we were given a graph G and had to find the largest clique in G. Recall that a clique is a set of nodes such that every node is connected to every other node in the set (i.e., everybody in the set knows everybody else in the set). The CLIQUE-3 problem is the same problem, except that we are *guaranteed* that every node has degree at most 3.

Suppose that I want to prove that CLIQUE-3 is NP-Complete. To do this, I make the following argument: We know that the CLIQUE problem is NP-Complete. CLIQUE-3 reduces to CLIQUE, because if we can solve CLIQUE for graphs in general, then clearly we can also solve it for graphs where the nodes have degree at most 3. Thus, CLIQUE-3 is also NP-Complete.

Part a: What is wrong with the above argument?

Part b: Show that CLIQUE-3 is *not* NP-Complete (assuming that P $\neq$ NP). (Hint: what is the largest possible clique in a graph where the nodes have degree at most 3?)

Solution for part a: The problem with this argument is that the reduction goes in the wrong direction: you would have to reduce CLIQUE to CLIQUE-3.
Solution for part b: There is a polynomial time algorithm to solve the CLIQUE-3 problem. If each node has degree at most 3, then the largest clique can be of size 4 at most. So iterate over all sets of 4 nodes (which takes O($|V|^4$) time) and all sets of 3 nodes (which takes O($|V|^3$) time), and see if a clique of size 3 or 4 exists. If none exist, then the largest clique has size 2 (a single edge).

Problem 4: In class, we considered the N Queens problem, where we had to place N queens on an N-by-N chessboard such that no two queens were in the same row, column, or diagonal (i.e., no two queens are threatening each other). We designed a simple local search algorithm for this problem as follows:

First, because we know that each column can contain only one queen, assign each queen to her own column. Place each queen at a random location within her column. Calculate the number of pairs of queens that are threatening each other (e.g., if Q_1 is threatened by both Q_2 and Q_3, this counts as 2 threats). For each queen Q_i, consider moving queen Q_i to a different location in the same column, and calculate the new number of conflicts that would exist if you performed that move. Once you have performed these calculations, perform the move that results in the greatest reduction in the current number of conflicts (subject to the constraint that each queen must remain in her assigned column). If there is no move that reduces the current number of conflicts, then terminate. Repeat until termination.

Give an example showing that this algorithm may fail to find a correct solution. (Hint: try $N = 4$. Show that (a) there is a solution in which all queens are safe, and (b) there is some starting position and some sequence of moves following the above description that fails to find such a solution.)

Solution: Suppose $N = 4$. Suppose the queens start off at positions (1, 1), (2, 2), (1, 3), (2, 4) in the chessboard (numbers indicate row and column positions). Let queen i be the queen in column i. Queen 3 has the most conflicts, and gets moved to (4, 1) so that all conflicts are removed. Now queen 2 has is in conflict with queens 1 and 4, so queen 2 needs to move. Any location will put her in conflict with 1 other queen, so randomly choose to move her to (3, 2). Now queens 2 and 3 are the only ones in conflict with each other, but any move will put them in conflict with another queen, so there is no way to get to the correct solution.

The other part to this solution is showing that there is a solution when $N = 4$. One such solution is (2, 1), (4, 2), (1, 3), (3, 4).